[Cf. Slide 7]

(a) A Mini-ML type checking problem:

$\boxed{\text{given closed } M \text{ and } \sigma, \text{ does } \{\} \vdash M : \sigma \text{ hold?}}$

(b) A Mini-ML typeability problem

$\boxed{\text{given closed } M, \text{ does there exist a closed } \sigma \text{ such that } \{\} \vdash M : \sigma \text{ holds?}}$

N.B. Solving (a) entails solving (b) because of the form of the (let) typing rule.

Two examples involving self-application

$$M \stackrel{\text{def}}{=} \texttt{let}\ f = \lambda x_1(\lambda x_2(x_1))\ \texttt{in}\ f\ f$$

$$M' \stackrel{\text{def}}{=} (\lambda f(f\ f))\ \lambda x_1(\lambda x_2(x_1))$$

Are M and M' typeable in the Mini-ML type system?

Figure 1 [p 19]

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{x_1 : \quad , x_2 : \quad \vdash x_1 :}\,(C3)
}{x_1 : \quad \vdash \lambda x_2(x_1) :}\,(C2)
}{\{\} \vdash \lambda x_1(\lambda x_2(x_1)) :}\,(C1)
\qquad
\dfrac{
\dfrac{}{f : \quad \vdash f :}\,(C5)
\quad
\dfrac{}{f : \quad \vdash f :}\,(C6)
}{f : \quad \vdash f f :}\,(C4)
}{\{\} \vdash \mathrm{let}\ f = \lambda x_1(\lambda x_2(x_1))\ \mathrm{in}\ f f \quad :}\,(C0)
$$

Figure 1 [p 19]

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 : \overset{\forall\{\}\tau_3}{\tau_3},\ x_2 : \overset{\forall\{\}\tau_5}{\tau_5} \vdash x_1 : \tau_6}\,(C3)}{x_1 : \tau_3 \vdash \lambda x_2(x_1) : \tau_4}\,(C2)}{\{\} \vdash \lambda x_1(\lambda x_2(x_1)) : \tau_2}\,(C1) \qquad \dfrac{\dfrac{}{f : \forall A(\tau_2) \vdash f : \tau_7}\,(C5) \quad \dfrac{}{f : \forall A(\tau_2) \vdash f : \tau_8}\,(C6)}{f : \forall A(\tau_2) \vdash f f : \tau_1}\,(C4)}{\{\} \vdash \text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f : \tau_1}\,(C0)$$

Constraints generated while inferring a type for $\texttt{let}\ f = \lambda x_1(\lambda x_2(x_1))\ \texttt{in}\ f\ f$

(C0) $A = ftv(\tau_2)$

(C1) $\tau_2 = \tau_3 \rightarrow \tau_4$

(C2) $\tau_4 = \tau_5 \rightarrow \tau_6$

(C3) $\forall\,\{\,\}\,(\tau_3) \succ \tau_6$, i.e. $\tau_3 = \tau_6$

(C4) $\tau_7 = \tau_8 \rightarrow \tau_1$

(C5) $\forall\,A\,(\tau_2) \succ \tau_7$

(C6) $\forall\,A\,(\tau_2) \succ \tau_8$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{matrix} \tau_6 = \alpha_1 \\ \tau_5 = \alpha_2 \end{matrix}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

$$\tau_2 \overset{(C1)}{=\!=} \tau_3 \to \tau_4 \overset{(C2)}{=\!=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=\!=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{array}{l} \tau_6 = \alpha_1 \\ \tau_5 = \alpha_2 \end{array}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

(C5) : $\forall\{\alpha_1, \alpha_2\}(\alpha_1 \to (\alpha_2 \to \alpha_1)) > \tau_7 \overset{(C4)}{=\!=} \tau_8 \to \tau_1$

(C6) : " " " $> \tau_8$

So $\begin{cases} \tau_8 \to \tau_1 = \tau_9 \to (\tau_{10} \to \tau_9) \\ \tau_8 = \tau_{11} \to (\tau_{12} \to \tau_{11}) \end{cases}$ for some $\tau_9, \tau_{10}, \tau_{11}, \tau_{12}$

$$\tau_2 \overset{(C1)}{=} \tau_3 \to \tau_4 \overset{(C2)}{=} \tau_3 \to (\tau_5 \to \tau_6) \overset{(C3)}{=} \tau_6 \to (\tau_5 \to \tau_6)$$

Take $\left.\begin{array}{l}\tau_6 = \alpha_1 \\ \tau_5 = \alpha_2\end{array}\right\}$ type variables.

So $A = ftv(\tau_2) = ftv(\alpha_1 \to (\alpha_2 \to \alpha_1)) = \{\alpha_1, \alpha_2\}$

(C5) : $\forall\{\alpha_1,\alpha_2\}(\alpha_1 \to (\alpha_2 \to \alpha_1)) > \tau_7 \overset{(C4)}{=} \tau_8 \to \tau_1$

(C6) : " " " $> \tau_8$

So $\begin{cases} \tau_8 = \tau_9 \ \& \ \tau_1 = (\tau_{10} \to \tau_9) \\ \tau_8 = \tau_{11} \to (\tau_{12} \to \tau_{11}) \end{cases}$ for some $\tau_9, \tau_{10}, \tau_{11}, \tau_{12}$

So $\tau_1 = \tau_{10} \to \tau_9 = \tau_{10} \to \tau_8 = \tau_{10} \to (\tau_{11} \to (\tau_{12} \to \tau_{11}))$

Thus

$$\{\} \vdash \left(\text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f\right) : \tau_{10} \to (\tau_{11} \to (\tau_{12} \to \tau_{11}))$$

holds for any $\tau_{10}, \tau_{11}, \tau_{12}$

So

$$\boxed{\begin{aligned} &\vdash \left(\text{let } f = \lambda x_1(\lambda x_2(x_1)) \text{ in } f f\right) : \\ &\qquad \forall \alpha_1, \alpha_2, \alpha_3\, (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_2))) \end{aligned}}$$

Two examples involving self-application

$$M \stackrel{\text{def}}{=} \texttt{let } f = \lambda x_1(\lambda x_2(x_1)) \texttt{ in } f\ f$$

$$M' \stackrel{\text{def}}{=} (\lambda f(f\ f))\ \lambda x_1(\lambda x_2(x_1))$$

Are M and M' typeable in the Mini-ML type system?

[Page 21]

The constraints generated from trying to type

$$(\lambda f(ff))\,\lambda x_1(\lambda x_2(x_1))$$

give

$$\tau_7 \overset{(C13)}{=} \tau_4 \overset{(C12)}{=} \tau_6 \overset{(C11)}{=} \tau_7 \to \tau_5$$

The constraints generated from trying to type

$$(\lambda f(ff))\ \lambda x_1(\lambda x_2(x_1))$$

give

$$\tau_7 \overset{(C13)}{=} \tau_4 \overset{(C12)}{=} \tau_6 \overset{(C11)}{=} \tau_7 \to \tau_5$$

these cannot be equal – they have different numbers of the symbol "→" in them

Principal type schemes for closed expressions

A closed type scheme $\forall\, A\,(\tau)$ is the *principal* type scheme of a closed Mini-ML expression M if

(a) $\vdash M : \forall\, A\,(\tau)$

(b) for any other closed type scheme $\forall\, A'\,(\tau')$,
if $\vdash M : \forall\, A'\,(\tau')$, then $\forall\, A\,(\tau) \succ \tau'$

eg $\forall \alpha_1, \alpha_2, \alpha_3\, (\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_2)))$ is principal type scheme for let $f = \lambda x_1 (\lambda x_2 (x_1))$ in $f\,f$

Theorem (Hindley; Damas-Milner)

If the closed Mini-ML expression M is typeable (i.e. $\vdash M : \sigma$ holds for some type scheme σ), then there is a principal type scheme for M.

Indeed, there is an algorithm which, given any M as input, decides whether or not it is typeable and returns a principal type scheme if it is.

An ML expression with a principal type scheme hundreds of pages long

$$
\begin{array}{l}
\texttt{let}\ \mathit{pair} = \lambda x(\lambda y(\lambda z(z\,x\,y)))\ \texttt{in} \\
\quad \texttt{let}\ x_1 = \lambda y(\mathit{pair}\ y\ y)\ \texttt{in} \\
\quad\quad \texttt{let}\ x_2 = \lambda y(x_1(x_1\,y))\ \texttt{in} \\
\quad\quad\quad \texttt{let}\ x_3 = \lambda y(x_2(x_2\,y))\ \texttt{in} \\
\quad\quad\quad\quad \texttt{let}\ x_4 = \lambda y(x_3(x_3\,y))\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{let}\ x_5 = \lambda y(x_4(x_4\,y))\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad x_5(\lambda y(y))
\end{array}
$$

(Taken from Mairson 1990.)

Principal type schemes for open expressions

A *solution* for the typing problem $\Gamma \vdash M : ?$ is a pair $\boxed{(S, \sigma)}$ consisting of a type substitution S and a type scheme σ satisfying

$$S\,\Gamma \vdash M : \sigma$$

(where $S\,\Gamma = \{x_1 : S\,\sigma_1, \ldots, x_n : S\,\sigma_n\}$, if $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$).

Such a solution is *principal* if given any other, (S', σ'), there is some $T \in \mathbf{Sub}$ with $T\,S = S'$ and $T(\sigma) \succ \sigma'$.

[For type schemes σ and σ', with $\sigma' = \forall\, A'\,(\tau')$ say, we define $\boxed{\sigma \succ \sigma'}$ to mean $A' \cap \mathit{ftv}(\sigma) = \{\}$ and $\sigma \succ \tau'$.]

Example typing problem:

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x\ \mathrm{true}\ ?}$$

Has solutions

$$S_1 = \{\beta \mapsto \mathrm{bool}\},\ \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$$

Example typing problem:

$$\boxed{x : \forall\alpha(\beta \to (\gamma \to \alpha)) \vdash x \text{ true } ?}$$

Has solutions

$$S_1 = \{\beta \mapsto \text{bool}\},\ \sigma_1 = \forall\alpha(\gamma \to \alpha)$$

BOTH PRINCIPAL SOLUTIONS

$$S_2 = \{\beta \mapsto \text{bool}, \gamma \mapsto \alpha\},\ \sigma_2 = \forall\alpha'(\alpha \to \alpha')$$

Example typing problem:

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x \text{ true } ?}$$

Has solutions

$S_1 = \{\beta \mapsto \text{bool}\}$, $\sigma_1 = \forall\alpha\,(\gamma \to \alpha)$

BOTH PRINCIPAL SOLUTIONS

$S_2 = \{\beta \mapsto \text{bool}, \gamma \mapsto \alpha\}$, $\sigma_2 = \forall\alpha'\,(\alpha \to \alpha')$

$S_3 = \{\beta \mapsto \text{bool}, \gamma \mapsto \alpha\}$, $\sigma_3 = \forall\alpha'\,(\alpha \to (\alpha' \to \alpha''))$

Example typing problem:

$$\boxed{x : \forall\alpha\,(\beta \to (\gamma \to \alpha)) \vdash x \text{ true } ?}$$

Has solutions

$S_1 = \{\beta \mapsto bool\},\ \sigma_1 = \forall\alpha\,(\gamma \to \alpha)$

BOTH PRINCIPAL SOLUTIONS

$S_2 = \{\beta \mapsto bool, \gamma \mapsto \alpha\},\ \sigma_2 = \forall\alpha'\,(\alpha \to \alpha')$

$S_3 = \{\beta \mapsto bool, \gamma \mapsto \alpha\},\ \sigma_3 = \forall\alpha'\,(\alpha \to (\alpha' \to \alpha''))$

$S_4 = \{\beta \mapsto bool, \gamma \mapsto bool\},\ \sigma_4 = \forall\{\}\,(bool \to bool)$

Properties of the Mini-ML typing relation

- If $\Gamma \vdash M : \sigma$, then for any type substitution $S \in \mathbf{Sub}$ $S\Gamma \vdash M : S\sigma$.
- If $\Gamma \vdash M : \sigma$ and $\sigma \succ \sigma'$, then $\Gamma \vdash M : \sigma'$.

Specification for the principal typing algorithm, pt

pt operates on typing problems $\Gamma \vdash M : ?$ (consisting of a typing environment Γ and an Mini-ML expression M). It returns either a pair (S, τ) consisting of a type substitution $S \in \mathrm{Sub}$ and an Mini-ML type τ, or the exception $FAIL$.

- If $\Gamma \vdash M : ?$ has a solution (cf. Slide 27), then $pt(\Gamma \vdash M : ?)$ returns (S, τ) for some S and τ;
  moreover, setting $A = (ftv(\tau) - ftv(S\,\Gamma))$, then $(S, \forall\, A\,(\tau))$ is a principal solution for the problem $\Gamma \vdash M : ?$.

- If $\Gamma \vdash M : ?$ has no solution, then $pt(\Gamma \vdash M : ?)$ returns $FAIL$.

Unification of ML types

There is an algorithm mgu which when input two Mini-ML types τ_1 and τ_2 decides whether τ_1 and τ_2 are *unifiable*, i.e. whether there exists a type-substitution $S \in \mathbf{Sub}$ with

(a) $S(\tau_1) = S(\tau_2)$.

Moreover, if they are unifiable, $mgu(\tau_1, \tau_2)$ returns the *most general unifier*—an S satisfying both (a) and

(b) for all $S' \in \mathbf{Sub}$, if $S'(\tau_1) = S'(\tau_2)$, then $S' = TS$ for some $T \in \mathbf{Sub}$.

By convention $mgu(\tau_1, \tau_2) = FAIL$ if (and only if) τ_1 and τ_2 are not unifiable.

Some of the clauses in a definition of pt

Function abstractions: $pt(\Gamma \vdash \lambda x(M) : ?) \stackrel{\text{def}}{=}$

$$\text{let } \alpha = \text{fresh in}$$
$$\text{let } (S, \tau) = pt(\Gamma, x : \alpha \vdash M : ?) \text{ in } (S, S(\alpha) \to \tau)$$

Function applications: $pt(\Gamma \vdash M_1\, M_2 : ?) \stackrel{\text{def}}{=}$

$$\text{let } (S_1, \tau_1) = pt(\Gamma \vdash M_1 : ?) \text{ in}$$
$$\text{let } (S_2, \tau_2) = pt(S_1\, \Gamma \vdash M_2 : ?) \text{ in}$$
$$\text{let } \alpha = \text{fresh in}$$
$$\text{let } S_3 = mgu(S_2\, \tau_1, \tau_2 \to \alpha) \text{ in } (S_3 S_2 S_1, S_3(\alpha))$$

Mini-ML type system, III

(**fn**) $$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Gamma \vdash \lambda x(M) : \tau_1 \to \tau_2}$$ if $x \notin dom(\Gamma)$

(**app**) $$\frac{\Gamma \vdash M_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash M_2 : \tau_1}{\Gamma \vdash M_1\, M_2 : \tau_2}$$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

$S_1 \Gamma \vdash M_1 : \tau_1$

$pt(\Gamma' \vdash M_1 M_2 : ?) =$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

$pt(S_1\Gamma \vdash M_2 : ?) = (S_2, \tau_2)$

+slide 28

$S_2 S_1 \Gamma \vdash M_1 : S_2 \tau_1$

$S_2 S_1 \Gamma' \vdash M_2 : \tau_2$

$pt(\Gamma' \vdash M_1 M_2 : ?) =$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

$pt(S_1\Gamma \vdash M_2 : ?) = (S_2, \tau_2)$

$mgu(S_2\tau_1, \tau_2 \to \alpha) = S_3$

+ Slide 28

$S_3\tau_2 \to S_3\alpha$

$\parallel$

$S_3S_2S_1\Gamma \vdash M_1 : S_3S_2\tau_1$

$S_3S_2S_1\Gamma' \vdash M_2 : S_3\tau_2$

$\hookrightarrow pt(\Gamma' \vdash M_1M_2 : ?) =$

$(S_3S_2S_1, S_3\alpha)$

$pt(\Gamma \vdash M_1 : ?) = (S_1, \tau_1)$

$pt(S_1\Gamma \vdash M_2 : ?) = (S_2, \tau_2)$

$mgu(S_2\tau_1, \tau_2 \to \alpha) = S_3$

$$\dfrac{S_3S_2S_1\Gamma' \vdash M_1 : \overset{S_3S_2\tau_1}{\overset{\|}{S_3\tau_2 \to S_3\alpha}} \qquad S_3S_2S_1\Gamma' \vdash M_2 : S_3\tau_2}{S_3S_2S_1\Gamma' \vdash M_1M_2 : S_3\alpha}\,(app)$$

$\hookrightarrow pt(\Gamma' \vdash M_1M_2 : ?) =$
$(S_3S_2S_1, S_3\alpha)$